REMARKS

Claims 1-27 are pending in the present application. The Office Action mailed April 24, 2002 (hereinafter "Office Action"), rejected Claims 1-27. More specifically, Claims 1-27 were rejected under 35 U.S.C. § 112, first paragraph, as containing subject matter that was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor, at the time the application was filed, had possession of the claimed invention. Claims 1-3, 8-11, 15, 16, 19, 22, 23, and 27 were rejected under 35 U.S.C. § 102(e) as being anticipated by U.S. Patent No. 6,105,036, issued to Henckel. Claims 12-14, 21, and 24-26 were rejected under 35 U.S.C. § 103(a) as being unpatentable over Henckel in view of U.S. Patent No. 5,784,553, issued to Kolawa et al. (hereinafter "Kolawa"). Claims 1, 4-7, 15, and 17-20 were rejected under 35 U.S.C. § 103(a) as being unpatentable over U.S. Patent No. 6,157,933 issued to Celi, Jr. et al. (hereinafter "Celi") in view of a nonpatent reference titled *Java 1.1*, Second Edition, by Jaworski (hereinafter "Jaworski"). Additionally, the Office Action objected to a typographical error in the specification at page 11, line 28, which is being corrected in this amendment. Applicant respectfully traverses the rejections set forth in the Office Action.

In this amendment, applicant is amending independent Claims 1, 8, 10, 15, and 19 for clarification purposes to more particularly point out and distinctly claim the subject matter that applicant regards as being his invention. More specifically, applicant is amending Claims 1, 8, 10, 15, and 19 to more particularly point out and distinctly claim subject matter regarding constructing the complex data object directly from the persistent representation, without parsing the persistent representation. Applicant is also adding new Claims 28-65 to more particularly point out and distinctly claim the subject matter that applicant regards as being his invention. Applicant respectfully submits that none of the cited and applied references, either alone or in combination, teach or suggest the subject matter recited in applicant's Claims 1-65.

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS^PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100

Before discussing in detail the reasons why applicant believes that Claims 1-65 are allowable, brief descriptions of the present invention and the cited and applied references are presented. The following discussion of the disclosed embodiments of applicant's invention and the discussion of the differences among the disclosed embodiments and the teachings in the applied references are not provided to define the scope or interpretation of any of the claims. Instead, such discussed differences are provided to help the United States Patent and Trademark Office better appreciate important claim distinctions discussed thereafter.

I. Summary of the Invention

The present invention is generally directed to providing a method, system, and storage medium for recreating a complex data object from a persistent representation of the structure of the complex data object. The persistent representation includes a sequence of directly executable instructions that are used to recreate the structure of the complex data object. The directly executable instructions are calls to a library of predefined functions. The complex data object is constructed directly from the persistent representation by calling the predefined functions that correspond to the sequence of directly executable instructions. No parsing of the persistent representation is performed in constructing the complex data object. The complex data object is constructed by a purely linear sequential execution of the persistent representation.

In accordance with another aspect of the present invention, a method, system, and storage medium for recreating a complex data object having a structure is provided. A sequence of calls from an authoring tool to a set of predefined functions for creating a complex data object are recorded and translated into a sequence of directly executable instructions. The sequence of directly executable instructions is stored as a persistent representation of the structure of the complex data object. To recreate the complex data object, the computer obtains the persistent representation of the structure of the complex data object and interprets the directly executable instructions as calls to a set of predefined functions. The computer calls a predefined function corresponding to each directly executable instruction in the sequence of directly executable instructions to construct the complex data object directly from the persistent representation.

Significant advantages result from the unique methods, systems, and storage media provided by the present invention for recreating complex data objects. By providing a persistent representation of the structure of the complex data object as a sequence of directly executable instructions that are calls to a set of predefined functions, the complex data object can be constructed directly from the persistent representation, without parsing the persistent representation. Thus, the present invention provides significant advantages by increasing the processing speed and reducing the strain on computer memory and processing resources.

II. Summary of Henckel

Henckel is generally directed toward disclosing a tool for assisting the user in developing multimedia presentation applications. Henckel discloses displaying a source code file with an ordered arrangement of object definitions of multimedia objects. The multimedia object definitions can selectively be displayed in either textual or multimedia representations in response to user input. The representations are inline within the ordered arrangement of object definitions, such that a visual indication of the arrangement of such object definitions in the source code file is maintained. Henckel discloses the use of conventional interpretative program statements and definitions that require parsing. Henckel generates multimedia representations by parsing the source code and forming a parse data structure, such as a tree.

Henckel fails to teach or suggest constructing a complex data object directly from a persistent representation of the structure of the complex data object as a sequence of directly executable instructions that are calls to predefined functions, without parsing the persistent representation. Only the present invention constructs a complex data object by a purely linear sequential execution of the persistent representation.

III. Summary of Celi

Celi is generally directed toward disclosing a method and apparatus for downloading multiple animated images on a Web page during browsing of a network with limited throughput. Celi discloses a method in which a Web page and associated Java applet with a default image are loaded. The loaded Java applet then displays the default image. After displaying the default image, the Java applet requests the Web server to deliver an image series, which is a list of related images. Once the first image in the series is downloaded, the Java animation applet prepares the image for display. The Java applet displays screen transition effects between the downloading of each subsequent image in the image series list. This creates the perception to the user of not waiting for additional information to download in the background. Celi is limited to disclosing conventional Java applets for displaying objects.

Celi fails to teach or suggest constructing a complex data object directly from a persistent representation of the structure of the complex data object as a sequence of directly executable instructions that are calls to predefined functions, without parsing the persistent representation. Only the present invention constructs a complex data object by a purely linear sequential execution of the persistent representation.

IV. Summary of Jaworski

Jaworski generally discloses several examples of Java applets to illustrate different aspects of Java classes and methods. Jaworski discloses a Java applet that shows how to load and play audio files and images. Jaworski also discloses Java applets that may be implemented to perform functions such as displaying text, video, or images in a browser application. Jaworski is limited to disclosing conventional Java applets for displaying objects.

Jaworski fails to teach or suggest constructing a complex data object directly from a persistent representation of the structure of the complex data object as a sequence of directly executable instructions that are calls to predefined functions, without parsing the persistent representation. Only the present invention constructs a complex data object by a purely linear sequential execution of the persistent representation.

V. Summary of Kolawa

Kolawa is generally directed toward disclosing a method and system for generating a test suite using dynamic symbolic execution for a computer program written in the Java programming language. Kolawa discloses finding an input for causing a program element, such as a program statement, to be executed in a manner that is extendable to finding a minimal set of inputs for executing substantially every statement at least once. Kolawa discloses a Java virtual machine that utilizes stacks to hold variables. Kolawa also discloses conventional interpretive Java program instructions that are represented as nodes in parse tree data structure. Kolawa discloses generating a test suite for a computer program using a dynamic symbolic execution interpreter that iteratively processes the instructions comprising the program under test using a parse tree representation, such as the parse tree.

Kolawa fails to teach or suggest constructing a complex data object directly from a persistent representation of the structure of the complex data object as a sequence of directly executable instructions that are calls to predefined functions, without parsing the persistent representation. Kolawa is limited to disclosing conventional interpreter that iteratively processes instructions using a parse tree. Only the present invention constructs a complex data object by a purely linear sequential execution of the persistent representation.

VI. Rejection of Claims 1-27 Under 35 U.S.C. § 112, First Paragraph

The Office Action rejected Claims 1-27 under 35 U.S.C. § 112, first paragraph, as containing subject matter that was not described in the specification in such a way as to reasonably convey to one skilled in the relevant art that the inventor, at the time the application was filed, had possession of the claimed invention. More specifically, the Office Action cites pages 7 and 10 of applicant's specification as indicating that the inventor, at the time the application was filed, did not have possession of constructing the complex data object directly from the persistent representation, "without parsing and translating an explicit definition of the complex data object structure." Applicant respectfully disagrees for the reasons discussed below.

In this amendment, applicant is amending independent Claims 1, 8, 10, 15, and 19 for clarification purposes to more particularly point out and distinctly claim the subject matter that applicant regards as being his invention. More specifically, applicant is amending Claims 1, 8, 10, 15, and 19 to more particularly point out and distinctly claim that the complex data object is constructed directly from the persistent representation, "without parsing the persistent representation." Applicant's specification clearly describes obtaining a persistent representation of the structure of the complex data object as a sequence of directly executable instructions. Applicant's specification also clearly describes calling a predefined function corresponding to each directly executable instruction in the sequence of directly executable instructions, so as to construct the complex data object directly from the persistent representation. Applicant's specification also clearly teaches that constructing the complex data object directly from the persistent representation requires "no parsing or conversion from a descriptive to an executable form." (Specification at page 8, lines 20-21.)

In a previous amendment, applicant added the limitation "without parsing and translating an explicit definition of the complex data object's structure," for clarification purposes only. In this amendment, applicant is removing this limitation. However, applicant respectfully disagrees with the Office Action's assertion that this subject matter was not described in the specification for several reasons. One reason is that even though, as the Office Action states, the specification describes translating a file containing code fragments to form a program, this is not the same as describing translating the code fragments to construct the complex data object. Rather, the complex data object is constructed directly from the program. Nowhere in applicant's specification is there any teaching or suggestion of parsing and translating an explicit definition of the complex data object's structure, so as to construct the complex data object. In contrast, the cited portion of page 7 of applicant's specification clearly states that "program generator 220 modifies, assembles, and translates the code fragments from file 212 to form a program 221" and that "instead of explicitly specifying the actual structure of the presentation, program 221 represents the structure as a program containing a sequence of instructions for recreating that structure." (Specification at page 7, lines 4-8.) Again, translating the file containing the entire sequence of code fragments to generate the program is not the same as translating the code fragments so as to construct the complex data object.

Additionally, applicant respectfully disagrees with the Office Action's assertion that constructing the complex data object directly from the persistent representation, "without parsing and translating an explicit definition of the complex data object structure," is not described in the specification because the specification describes a series of steps that traverse a tree and suggest parsing and translating. While the specification describes steps that traverse a tree, the steps are not performed so as to construct the complex data object. Rather the steps are performed to generate instructions for each code fragment to form a program. The cited portion of applicant's specification references FIGURE 4 and describes block 400, which "generates instructions for the virtual machine." The cited portion of applicant's specification clearly states that "this series of steps traverses a tree equivalent to a list, generating instructions for each fragment in an order that is compatible with the target virtual machine." (Specification at page 10, lines 23-25.) Nowhere does applicant's specification teach or suggest constructing a complex data object by parsing and translating an explicit definition of the complex data object's structure. In contrast, applicant's specification clearly teaches constructing a complex data object directly from the persistent representation that represents the structure of the complex data object as a sequence of directly executable instructions.

For the reasons discussed above, applicant respectfully submits that the subject matter recited in Claims 1-27 is described in the specification so as to reasonably convey to one skilled in the relevant art that the inventor, at the time the application was filed, had possession of the claimed invention. Therefore, applicant respectfully requests that the section 112, first paragraph, rejection be withdrawn and Claims 1-27 be allowed to issue.

VIII. Rejection of Claims 1-3, 8-11, 15, 16, 19, 22, 23, and 27 Under 35 U.S.C. § 102(e)

The Office Action rejected Claims 1-3, 8-11, 15, 16, 19, 22, 23, and 27 under Section 102(e) as being anticipated by Henckel. More specifically, the Office Action states that Henckel teaches a source file containing an ordered arrangement of interpretative program statements and object definitions, which is displayed so that the object definitions can be in either textual or multimedia representations. The Office Action also states that Henckel teaches the display or presentation of multimedia objects, with regard to Claims 2, 3, 9, 11, and 16. Applicant respectfully disagrees for the reasons discussed below.

In this amendment, applicant is amending independent Claims 1, 8, 10, 15, and 19 for clarification purposes to more particularly point out and distinctly claim subject matter regarding "calling a predefined function corresponding to each directly executable instruction in the sequence of directly executable instructions so as to construct the complex data object directly from the persistent representation, *without parsing the persistent representation.*" Applicant respectfully submits that Henckel fails to teach or suggest calling a predefined function corresponding to each directly executable instruction in the sequence of directly executable instructions so as to construct the complex data object directly from the persistent representation, "*without parsing the persistent representation,*" as recited in Claims 1, 8, 10, 15, and 19.

In contrast to the present invention, as embodied in Claims 1, 8, 10, 15, and 19, Henckel is limited to teaching the use of conventional interpretative program statements and definitions that explicitly specify the structure of the multimedia object and require parsing. Henckel fails to teach or suggest constructing a complex data object by a purely linear sequential execution of the persistent representation. Rather, Henckel generates multimedia representations by parsing the source code and forming a parse data structure, such as a tree.

> Browser 34 typically operates by *parsing the source code provided by main block 31 to form a parse data structure such as a tree*, and then utilizing the data structure to generate graphical and/or audio data for output to display 22 and audio system 29, in a manner that is well understood in the art. (Col. 6, lines 31-36, emphasis added.)

Since Henckel relies on the use of conventional static data structures parsed from source code, applicant asserts that Henckel fails to teach or suggest obtaining a persistent representation of a complex data object as a sequence of directly executable instructions and constructing the complex data object directly from the persistent representation, "without parsing the persistent representation." Henckel fails to teach or suggest constructing a complex data object by a purely linear sequential execution of the persistent representation.

Therefore, Henckel fails to teach or suggest applicant's invention as recited in the amended independent Claims 1, 8, 10, 15, and 19 and their respective dependent Claims 2, 3, 9, 11, 16, and 27. Thus, applicant respectfully submits that Claims 1-3, 8-11, 15, 16, 19, and 27 are allowable. Claims 22 and 23 are computer-readable-medium and system format claims, respectively, and parallel the methods of Claims 1-7. Therefore, the analysis discussed above with respect to independent Claim 1 also applies to Claims 22 and 23. Thus, applicant respectfully submits that Claims 22 and 23 are allowable for the same reasons as Claim 1. Therefore, applicant respectfully requests that the Section 102(e) rejection be withdrawn and Claims 1-3, 8-11, 15, 16, 19, 22, 23, and 27 be allowed to issue.

IX. Rejection of Claims 12-14, 21, and 24-26 Under 35 U.S.C. § 103(a)

The Office Action rejected dependent Claims 12-14, 21, and 24-26 under 35 U.S.C. § 103(a) as being unpatentable over Henckel in view of Kolawa. The Office Action cited Kolawa as disclosing an interpreter that is a local stack-based virtual machine including a temporary storage array. The Office Action stated that it would have been obvious to one of ordinary skill in the art, at the time the invention was made, to utilize the stack-based virtual machine taught by Kolawa in combination with the teachings of Henckel because Henckel recognized that source code files may be an interpreted language.

As discussed above, applicant is amending independent Claims 1, 8, 10, 15, and 19 for clarification purposes to more particularly point out and distinctly claim subject matter regarding "calling a predefined function corresponding to each directly executable instruction in the sequence of directly executable instructions so as to construct the complex data object directly from the persistent representation, *without parsing the persistent representation.*" As also discussed above, Henckel fails to teach or suggest constructing the complex data object directly from the persistent representation, "without parsing the persistent representation." Henckel fails to teach or suggest constructing a complex data object by a purely linear sequential execution of the persistent representation. Rather, Henckel is limited to teaching the use of conventional interpretive program statements and definitions that explicitly specify the structure of a multimedia object and require parsing. Henckel generates multimedia representations by parsing the source code and forming a parse data structure, such as a tree.

Similarly to Henckel, Kolawa is also limited to teaching conventional interpretive Java program instructions that are represented as nodes in parse tree data structure. Kolawa discloses generating a test suite for a computer program using a dynamic symbolic execution interpreter that iteratively processes the instructions comprising the program under test using a parse tree representation, such as the parse tree. For example, Kolawa describes generating a test suite for a computer program as follows:

> The dynamic symbolic execution interpreter 110 iteratively processes (blocks121-128) the instructions comprising the program under test using a *parse tree representation, such as the parse tree* shown in FIG. 2B. The dynamic symbolic execution *interpreter110 begins by reading the first instruction from the parse tree* (block 120). (Col. 12, lines 51-56, emphasis added.)

As quoted above, Kolawa is limited to disclosing conventional interpreter that iteratively processes instructions using a parse tree. Because Kolawa relies on the use of conventional static data structures parsed from source code, applicant asserts that Kolawa fails to teach or suggest obtaining a persistent representation of a complex data object as a sequence of directly executable instructions and constructing the complex data object directly from the persistent representation, "without parsing the persistent representation." Only the present invention constructs a complex data object by a purely linear sequential execution of the persistent representation.

Therefore, applicant respectfully submits that Henckel and Kolawa, either alone or in combination, fail to teach or suggest constructing a complex data object directly from the persistent representation, "without parsing the persistent representation," as recited in Claims 8, 10, and 19 and their respective dependent Claims 12-14, 21, and 24-26. Thus, for at least the same reasons discussed above with respect to independent Claims 8, 10, and 19, dependent Claims 12-14, 21, and 24-26 are allowable, and applicant respectfully requests the withdrawal of the Section 103(a) rejection of Claims 12-14, 21, and 24-26.

X. Rejection of Claims 1, 4-7, 15, 17-19, and 20 Under 35 U.S.C. § 103(a)

The Office Action rejected Claims 1, 4-7, 15, 17-19, and 20 under 35 U.S.C. § 103(a) as being unpatentable over Celi in view of Jaworski. More specifically, the Office Action stated that it would have been obvious to one of ordinary skill in the art at the time the invention was made to combine the examples of Java applets taught by Jaworski with the retrieval and display of applet-embedded Web pages taught by Celi because Celi teaches that applet content may be embedded. The Office Action additionally refers to Jaworski for teaching that some functions return an explicit result and some functions have arguments, which may further call another function or may be a constant value. The Office Action further states that one of ordinary skill in the art at the time of making applicant's invention would have known that method resolution between similarly named functions in object-oriented programming may be facilitated through argument matching. Applicant respectfully disagrees, for the following reasons.

As discussed above, applicant is amending independent Claims 1, 8, 10, 15, and 19 for clarification purposes to more particularly point out and distinctly claim subject matter regarding "calling a predefined function corresponding to each directly executable instruction in the sequence of directly executable instructions so as to construct the complex data object directly from the persistent representation, *without parsing the persistent representation*." The complex data object is constructed by a purely linear sequential execution of the persistent representation. Claims 4-7 depend from amended Claim 1. Claims 17 and 18 depend from amended Claim 15, and Claim 20 depends from amended Claim 19. Therefore, all of Claims 1, 4-7, 15, 17-19, and 20 include subject matter regarding "calling a predefined function corresponding to each directly executable instruction in the sequence of directly executable instructions so as to construct the complex data object directly from the persistent representation, *without parsing the persistent representation*."

Applicant respectfully submits that both Celi and Jaworski fail to teach or suggest applicant's novel invention for "calling a predefined function corresponding to each directly executable instruction in the sequence of directly executable instructions so as to construct the complex data object directly from the persistent representation, without parsing the persistent representation." Both Celi and Jaworski are limited to describing the use of conventional Java applets for displaying multimedia objects. Conventional Java applets include interpretive Java program instructions represented as nodes in parse tree data structure. The Java applets described by Jaworski and the applets embedded in the Web pages described by Celi are conventional Java programs that can be downloaded over the Internet and executed on the recipient's machine. These types of conventional Java applets explicitly specify the structure of the multimedia objects and are parsed by the Java enabled browser for display.

For example, Celi describes "retrieving hypertext objects containing a Java applet from servers over a network into a browser utilizing a Java engine for running Java applets." (Celi, Abstract.) Similarly, Jaworski discloses nothing other than a few examples of simple conventional Java applets. For example, Jaworski discloses source code for a simple Java applet that displays the text "Hello Web!" to the Java enabled browser window. (Jaworski, pages 734-735.) Nowhere in either Celi nor Jaworski is there any teaching or suggestion of obtaining a persistent representation of the structure of the complex data object as a sequence of directly executable instructions, much less constructing the complex data object directly from a persistent representation, without parsing the persistent representation. Again, Celi and Jaworski are limited to disclosing the use of conventional Java applets for displaying objects. Conventional

Java interpreter instructions do not call "a predefined function corresponding to each directly executable instruction in the sequence of directly executable instructions so as to construct the complex data object directly from the persistent representation, without parsing the persistent representation." Only the present invention constructs a complex data object by a purely linear sequential execution of the persistent representation.

Therefore, applicant respectfully submits that Celi and Jaworski, either alone or in combination, fail to teach or suggest applicant's invention, as recited by Claims 1, 4-7, 15, 17-19, and 20. Thus, Claims 1, 4-7, 15, 17-19, and 20 are allowable and applicant respectfully requests that the Section 103(a) rejection be withdrawn and Claims 1, 4-7, 15, 17-19, and 20 be allowed to issue.

CONCLUSION

In view of the foregoing, applicant respectfully submits that all the claims of the present application, Claims 1-65, are allowable over the cited and applied references, alone or in combination. Reconsideration and reexamination of the application are requested and allowance of the rejected claims and passage of the application to issue at an early date are solicited. If the Examiner has any questions or comments concerning this application, he is invited to contact the applicant's undersigned attorney at the number set forth below.

CHRISTENSEN O'CONNOR
JOHNSON KINDNESS^PLLC

Barbara M. Level

Barbara M. Level
Registration No. 45,483
Direct Dial No. 206.695.1776

LAW OFFICES OF
CHRISTENSEN O'CONNOR JOHNSON KINDNESS^PLLC
1420 Fifth Avenue
Suite 2800
Seattle, Washington 98101
206.682.8100

O[illegible] C178
AUG 3 0 2002
PATENT & TRADEMARK OFFICE

I hereby certify that this Amendment C and Response is being deposited with the U.S. Postal Service in a sealed envelope as first class mail with postage thereon fully prepaid and addressed to the U.S. Patent and Trademark Office, P.O. Box 2327, Arlington, VA 22202, on the below date.

Date: 8/26/02 Carole Julyan

BML:ejh/skg

VERSION WITH MARKINGS TO SHOW CHANGES MADE AUGUST 26, 2002

In the Specification:

The paragraph beginning at page 11, line 20, has been amended as follows:

When all arguments of the current fragment have been processed, block 450 appends an instruction to the instruction stream that calls the function named in the current fragment. In TABLE I, this is the entry in the "Function Name" column, the name that had been derived ultimately from a name in the augmented API 211, Fig. 2. Methods written in an object-oriented programming language usually also require pushing the object for which the method is being called, because a stack-based virtual machine must know what object to call the function on, and expects to find this object on the top of the stack. For example, a call to `image.transform(arguments)` involves generating code that evaluates the arguments to transform onto the stack[.], then generating code that evaluates the image object onto the stack, and finally outputting the instruction that calls the function. Placing the object on the stack may employ other function calls. Static methods are not defined on objects, and thus do not require pushing an object onto the stack. Technically, the object for which a method is called is also an argument of the method; however, the term "argument" herein does not include such an object, unless explicitly stated otherwise. Java and C++ commonly refer to this extra argument as the implicit "this" object.

In the Claims:

1. (Three Times Amended) A computer-implemented method for recreating a complex data object having a structure, the method comprising:

obtaining a persistent representation of the structure of the complex data object as a sequence of directly executable instructions, wherein the directly executable instructions are calls to a set of predefined functions;

interpreting the directly executable instructions as calls to a set of predefined functions; and

calling a predefined function corresponding to each directly executable instruction in the sequence of directly executable instructions so as to construct the complex data object directly from the persistent representation, without parsing [and translating an explicit definition of the complex data object's structure] <u>the persistent representation</u>.

8. (Three Times Amended) A system for recreating a complex data object having a structure, the system comprising:

a persistent representation of the structure of the complex data object and containing a sequence of directly executable instructions, wherein the instructions are calls to a predefined set of data types and methods for creating complex data objects;

a library having the predefined set of data types and methods for creating complex data objects; and

a program interpreter for directly executing the instructions as a sequence of calls on the library so as to directly construct the complex data object from the persistent representation, without parsing [and translating an explicit definition of the complex data object's structure] <u>the persistent representation</u>.

10. (Three Times Amended) A system for recreating a complex data object from a persistent representation of its structure, the system comprising:

a library having a predefined set of data types and methods for creating complex data objects; and

a program interpreter for interpreting the contents of the persistent representation as a sequence of directly executable instructions, and for executing those instructions as a sequence of calls on the library so as to construct the complex data object directly from the persistent representation, without parsing [and translating an explicit definition of the complex data object's structure] <u>the persistent representation</u>.

15. (Twice Amended) A storage medium containing a persistent representation of the structure of a multicomponent data object as a sequence of instructions directly executable, wherein the directly executable instructions are calls to a set of predefined functions that are called by a program interpreter implemented in a digital computer so as to recreate the structure of the multicomponent data object, without parsing [and translating an explicit definition of the multicomponent data object's structure] <u>the persistent representation</u>.

19. (Three Times Amended) A storage medium containing computer-executable instructions and data for interpreting a persistent representation of the structure of a complex data object as a sequence of directly executable virtual instructions for directly constructing the complex data object as a series of calls on a library of predefined functions, without parsing [and translating an explicit definition of the complex data object's structure] <u>the persistent representation</u>.

Claims 28-65 have been added.